|  |  |  |
| --- | --- | --- |
| Logo__SSPU_2016_Barva | | |
| **Závěrečná studijní práce**  **dokumentace** | | |
| **Server/Klient objednávkový systém v C#** | | |
| Lukáš Moravec | | |
|  | | |
|  | |  |
| **Obor:** | 18-20-M/01 INFORMAČNÍ TECHNOLOGIE  se zaměřením na počítačové sítě a programování | |
| **Třída:**  **Školní rok:** | IT4  2017/2018 | |

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité   
informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové   
a umělecké škole v Opavě, Praskova 399/8.

V Opavě 31. 12. 2017

*podpis autora práce*

**ANOTACE**

Realizací projektu se stal systém předobjednávek restaurací rychlého občerstvení, který by zkrátil dobu čekání na samotné objednání či výdej jídla. Projekt se skládá ze serverové části, která využívá softwarové platformy Mono pro .NET Framework psané v jazyce C# a GTK# pro tvorbu uživatelského rozhraní, a klientské části. Klient má v sobě zabudovanou platební bránu, aby mohla být objednávka zaplacena předem. Byl vytvořen pomocí Ionic Frameworku, který využívá webové technologie k tvorbě hybridních, adaptivních mobilních aplikací. Informace o uživatelích a detailech objednávek jsou uloženy v databázovém systému MySQL. Součástí serveru je i možnost vytvořit si vlastní jídelní lístek s možností exportu. Uživatel na straně klienta si může vymyslet své vlastní menu z aktuální nabídky na serveru.

[Úvod 5](#_Toc501285902)

[1 Teoretická a metodická východiska 6](#_Toc501285903)

[1.1 Způsoby Internetové komunikace 6](#_Toc501285904)

[1.1.1 XML RPC 6](#_Toc501285906)

[1.1.2 SOAP 6](#_Toc501285907)

[1.1.3 REST 6](#_Toc501285908)

[1.2 MVVM architektura 6](#_Toc501285909)

[1.2.1 Model 6](#_Toc501285910)

[1.2.2 View 7](#_Toc501285911)

[1.2.3 View model 7](#_Toc501285912)

[2 Využité technologie 7](#_Toc501285913)

[2.1 Mono 8](#_Toc501285914)

[2.2 Entity Framework 8](#_Toc501285915)

[2.3 MySQL 8](#_Toc501285916)

[2.4 Ionic 8](#_Toc501285917)

[2.4.1 Typescript 8](#_Toc501285918)

[2.4.2 Sass 8](#_Toc501285919)

[2.4.3 HTML5 9](#_Toc501285920)

[2.4.4 AngularJS 9](#_Toc501285921)

[2.5 Braintree Direct 9](#_Toc501285922)

[3 popis aplikace – serverová část 11](#_Toc501285923)

[3.1 Adresářová struktura 11](#_Toc501285924)

[3.2 Databáze 12](#_Toc501285925)

[3.3 Návrhář menu 14](#_Toc501285926)

[4 Popis aplikace – Klientská část 15](#_Toc501285927)

[4.1 Adresářová struktura 15](#_Toc501285928)

[4.2 Vlastní menu 16](#_Toc501285929)

[4.3 Objednávky a platby 17](#_Toc501285930)

[Závěr 19](#_Toc501285931)

[Seznam použitýCH INFORMAČNÍCH ZDROJů 20](#_Toc501285932)

Úvod

V dnešním rychlém světě je snad každý rád, za jakoukoliv minutu, kterou může ušetřit prostřednictvím mobilu a internetového připojení. Možnost objednat si jídlo a nechat si ho dovézt již samozřejmě byla realizována. Ale co když jsem již na místě a nepotřebuji žádný dovoz a chci si rychle objednat jídlo bez čekání v řadě? Tato myšlenka mě vedla k realizaci nápadu, který by mohl ušetřit drahocenný čas.

Začal jsem uvažovat nad technologiemi, které by bylo vhodné použít. Chtěl jsem, aby byly aplikace multiplatformní a zároveň použít nějaké mi blízké řešení. Rozhodl jsem se tedy vytvořit serverovou MVVM aplikaci s použitím jazyka C# ve spojení s frameworkem .NET, aby bylo možné server spustit na jakémkoli operačním systému. Pro mobilní aplikaci (klienta) jsem použil Ionic Framework kvůli jednoduchému použití a taktéž multiplatformnímu řešení. Na ukládání různých dat padla má volba na využití databázového systému MySQL, protože ho dobře znám ze svého studia.

Nejvíce jsem se snažil vytvořit systém, který by jednoduše umožnil vytvářet a spravovat aktuální nabídku v restauraci. Tak vznikl můj „návrhář“ jídelního lístku. Jeho použitím jde jednoduše vytvářet skupiny jídel (např. snídaně, obědy, …) a třeba i další podskupiny (např. vegetariánská, …). Samozřejmostí je i možnost k uloženým pokrmům zadat i jejich výživové hodnoty, včetně alergenů, které obsahují. Všechny tyto informace si poté uživatel na mobilním zařízení může jednoduše zobrazit, učinit předobjednávku a zaplatit.

První část dokumentace je zaměřena na použité postupy, architektury a technologie. Druhá část postupně popisuje způsob fungování nejdříve serveru a poté klienta. Dále se Nachází ukázky uživatelského rozhraní s popisem funkčnosti.

# Teoretická a metodická východiska

## Způsoby internetové komunikace

## HTTP (neboli Hypertext Transfer Protocol) je internetový protokol určený pro výměnu hypertextových dokumentů. Samotný protokol HTTP neumožňuje šifrování ani zabezpečení integrity dat. Pro zabezpečení HTTP se často používá TLS spojení, společně označováno jako HTTPS. Za svou existenci vznikly různé možnosti jak využít tento protokol ke snadné výměně informací přes internet.

### XML RPC

XML Remote Procedure Call – protokol, který využívá syntaxe XML k volání metod uložených na jiném počítači, nezávisle na síti, přes HTTP.

### SOAP

Simple Object Access Protocol – nástupce XML RPC, poskytuje prostředí pro složitější komunikace.

### REST

Representational State Transfer – architektura pro vytváření aplikačních rozhraní pro distribuované prostředí. Používá se jako alternativa k RPC.

## MVVM architektura

MVVM je architektonický vzor pro tvorbu desktopových aplikací s grafickým uživatelským rozhraním. Skládá se z tří logických částí: model, bude a view-model

### Model

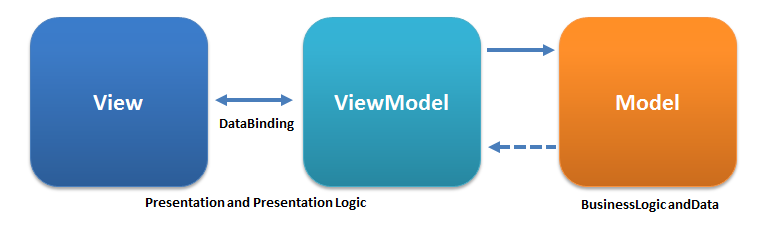
Vrstva přistupující k datům, která tvoří obsah aplikace. Na základě požadavků tyto data ukládá a vrací.

### View

Pojmem view se označuje grafický design, struktura, rozpoložení a vzhled toho, co uživatel vidí. Obvykle se jedná o nějaký XML kód grafických prvků. Například u WPF aplikací se jedná o XAML.

### View model

View model tvoří mezivrstvu mezi view a modelem. Má na starost zacházení s logikou view. S modelem spolupracuje tak, že invokuje metody v modelových třídách a získává data. Tyto data potom mohou být nějak formátovaná a předána do view pro zobrazení. Zároveň view model poskytuje všechny implementace příkazů, které uživatel iniciuje ve view. Například, pokud uživatel zmáčkne nějaké tlačítko v grafickém rozhraní, tak spustí se příkaz ve view modelu.



Obrázek . - Architektura MVVM

# Využité technologie

Technologie, které jsem použil k naprogramování tohoto projektu, jsem volil tak, aby byly spustitelné na co nejvíce zařízeních, tzv. multiplatformní. Díky tomuhle je možnost nasazení větší, než kdyby byl projekt závislý na jedné platformě.

## Mono

Mono je open source vývojová platforma založená na .NET Frameworku, pro tvorbu desktopových aplikací. Umožňuje kód zkompilovat na různých platformách. Součástí je integrace GTK# pro tvorbu grafického rozhraní.

## Entity Framework

ORM (object/relational mapping) systém Entity Framework umožňuje pomocí objektů vytvářet a přistupovat relační databáze. Zbavuje programátora nutnosti psát SQL příkazy náhradou za metody vracející potřebná data.

## MySQL

Databázový systém MySQL posloužil k ukládání dat. Mezi jeho přednosti patří to, že je multiplatformní a dobře zdokumentovaný.

## Ionic

Ionic Framework umožňuje tvorbu hybridních aplikací. Je vytvořen na systému Cordova. Využívá webových technologií a softwarových vývojových balíčků, díky kterým se aplikace dají jedním kódem zprovoznit na všech mobilních zařízeních a webu.

### Typescript

Jedná se o nadstavbu JavaScriptu. Přidává statické typování, třídy a rozhraní. Umožňuje jednodušší odchytávání chyb a celkově umožňuje psát robustnější aplikace.

### Sass

Preprocesor kaskádových stylů Sass má své výhody v možnosti používání proměnných, cyklů, dědičnosti, vnořených pravidel a základních matematických operací. Tyto soubory mívají příponu *.sass* nebo *.scss.*

### HTML5

Nejnovější standart značkovacího jazyku pro tvorbu webových stránek.

### AngularJS

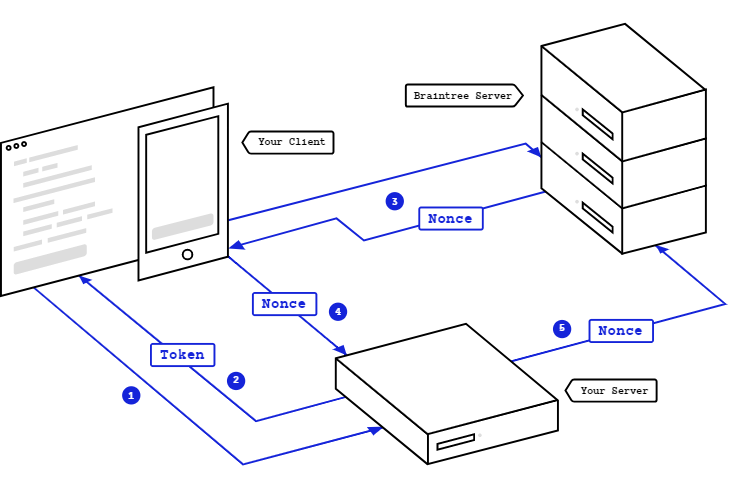
Jedná se o framework využívaný většinou pro dynamické webové aplikace. Umožňuje použít HTML jako šablonovací systém a vytvářet vlastní komponenty. Je schopný vázat data na jednotlivé elementy a značně usnadňuje psaní kódu.

## Braintree Direct

Platební brána Braintree Direct řeší zabezpečené mobilní platby a to pomocí:

* platebních karet,
* PayPalu,
* ApplePay.

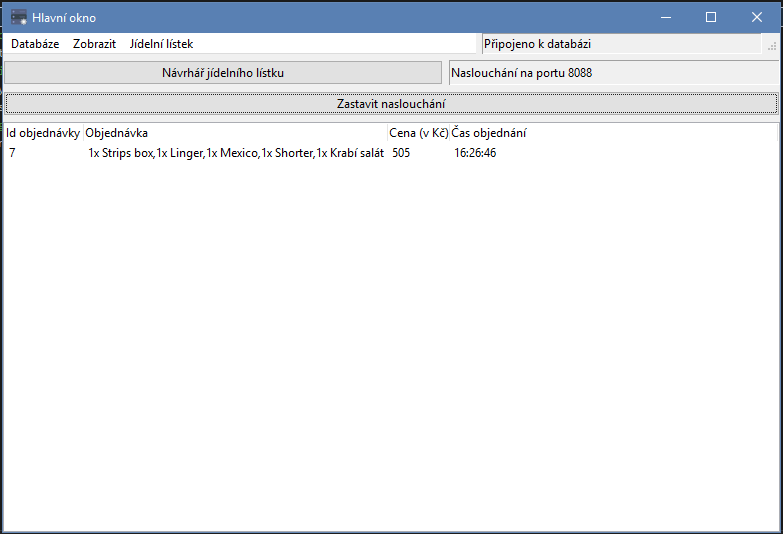
Pro každého uživatele se na serveru Braintree vytvoří záznam o zákazníkovi. Fungování je následovné:



Obrázek . – Způsob fungování Braintree

1. Klient pošle na můj server požadavek o svůj token, aby mohl inicializovat SDK.
2. Server odešle token zpátky s informacemi o jeho SDK.
3. Jakmile se SDK inicializuje a uživatel schválí své platební informace, SDK komunikuje se serverem Braintree a dostane *nonce*[[1]](#footnote-1) platební metody.
4. Poté klient odešle *nonce* zpátky na můj server.
5. Server *nonce* zpracuje a provede transakci ve spolupráci se serverem Braintree.

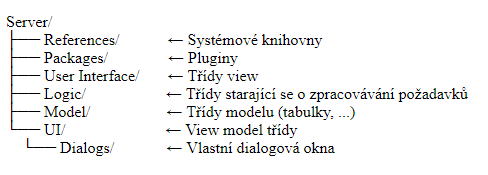
# popis aplikace – serverová část

Server, jak jsem již psal, je napsán v jazyce C# s využitím Mono frameworku, který zaručuje spustitelnost na více platformách. Využívá třídy HttpListener k naslouchání požadavků a tyto požadavky poté zpracovává asynchronně, aby nedocházelo k příliš velkým prodlevám.

Obrázek . - Server

## Adresářová struktura

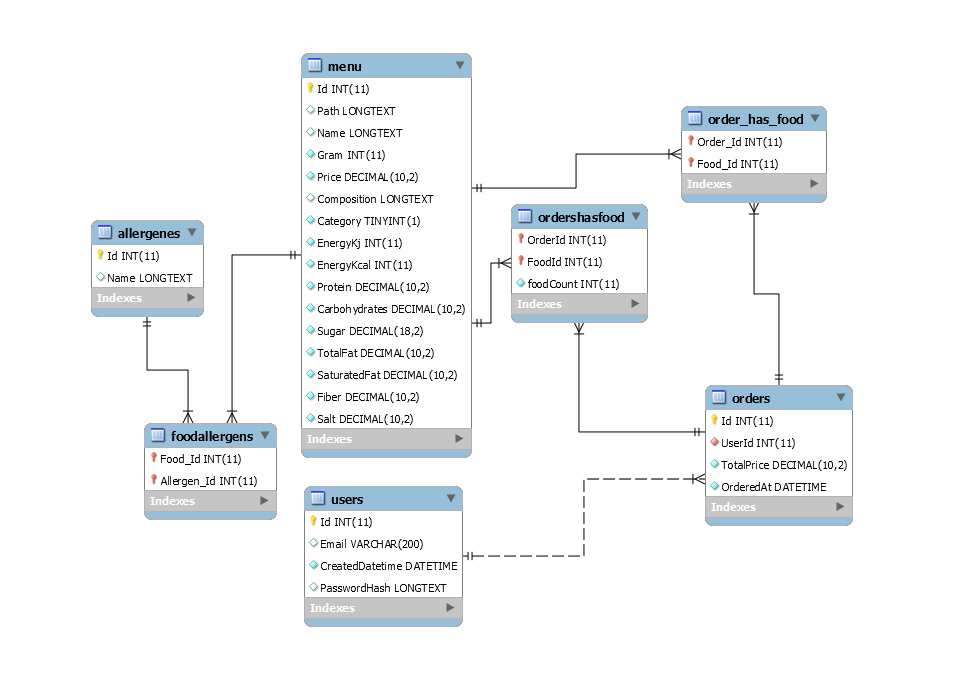
Adresářovou strukturu si při psaní desktopových aplikací většinou programátor vymýšlí sám a není předen daná, jako např. u webových MVC aplikací. Proto jsem se snažil vytvořit takovou strukturu, která dává nějaký smysl. Výsledek je v následujícím obrázku.



Obrázek . - Adresářová struktura serveru

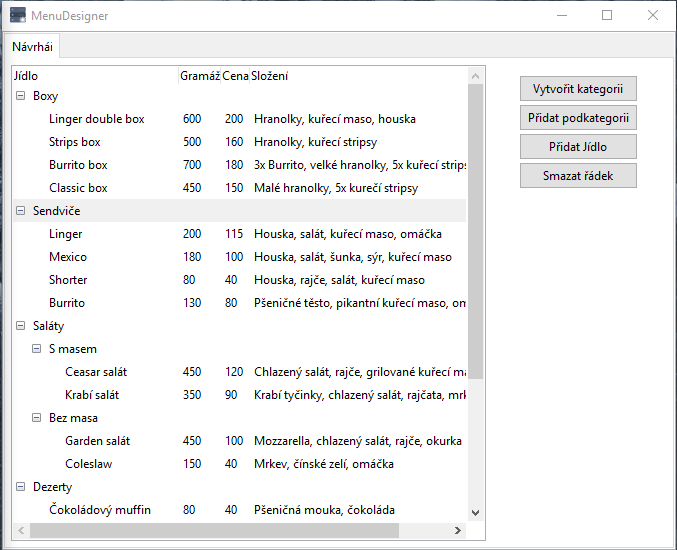
## Databáze

Databáze v MySQL je celá vygenerována automaticky. O to se stará zmíněný Entity Framework. Ten databázi vygeneruje na základě pojmenování jednotlivých tříd a jejich vlasností. Názvy tabulek se však dají změnit pomocí datových anotací. Ty můžeme využít i ke určení primárních a cizích klíčů, délky datových typů, indexů a nebo určit, které vlasnosti nechceme mapovat.

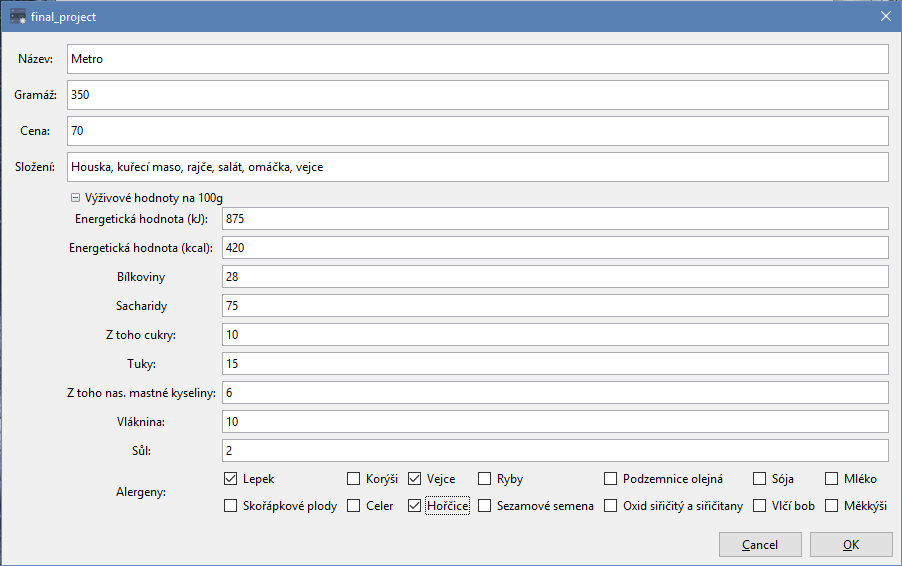


Obrázek . - Databázový model

## Návrhář menu

Návrhář menu je grafické okno, ve kterém mohou správci vytvořit a upravovat jídelní lístek ve formě stromu. Mohou se vytvářet kategorie a podkategorie jídel. U každého jídla se dají určit hlavní výživové hodnoty včetně alergenů.

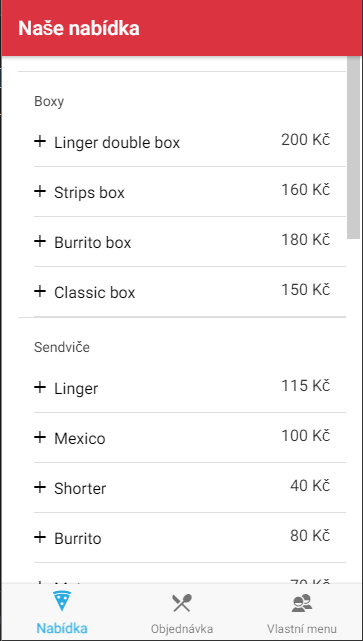
Obrázek . Návrhář menu



Obrázek . - Přidání potraviny

# Popis aplikace – Klientská část

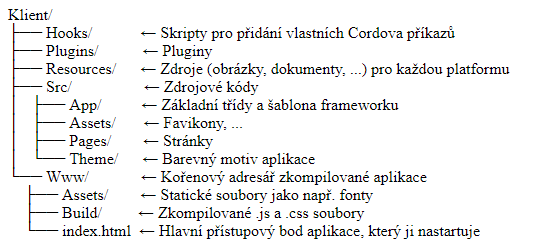
Po přihlášení se uživateli zobrazení aktuální nabídka, členěná do kategorií. Zobrazí se název kategorie, názvy položek, podkategorie a cena. Data k zobrazení se získají pomocí požadavku na server, který je vytáhne z databáze a pošle zpět. Mobilní aplikace má tři hlavní stránky, mezi kterýma se dá jednoduše přepínat. A to pro zobrazení nabídky, objednávky a vytvoření vlastního menu.



Obrázek . - Zobrazení nabídky v klientovi

## Adresářová struktura

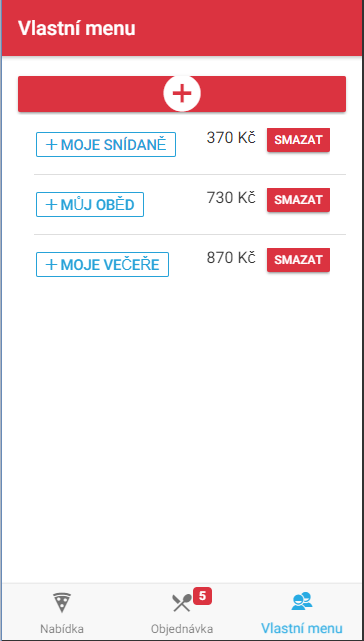
Na rozdíl od serveru, klient již má danou základní adresářovou strukturu frameworkem. Aplikace se spustí souborem index.html, který obsahuje odkazy pro zkompilovaný javascript a kaskádové styly. Adresář *pages* obsahuje všechny stránky. Každá stránka má svůj vlastní adresář, ve kterém se nachází skriptový soubor s třídou ve formátu *.ts*, šablona *.html* a pro stránku vlastní styl *.scss.*



Obrázek . - Adresářová struktura Ionic projektu

## Vlastní menu

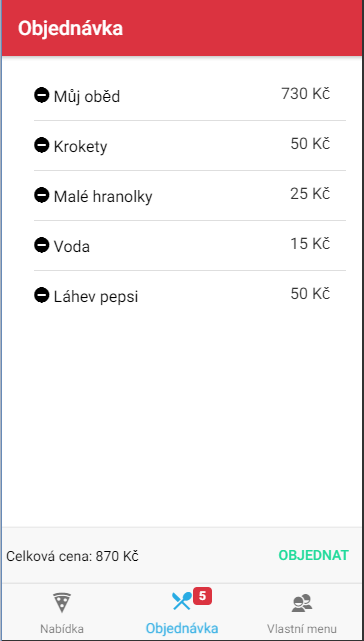
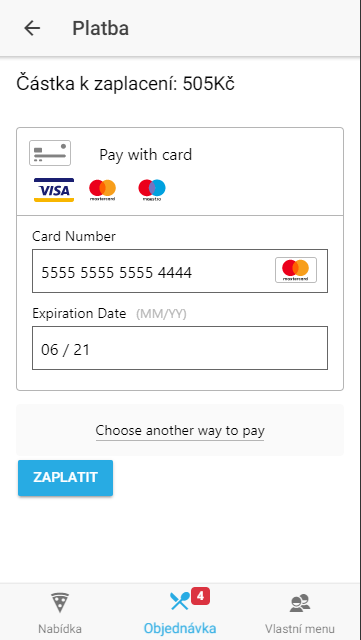
Každý uživatel si může sestavit své vlastní menu dle vlastní libosti. Stačí pár kliknutí, vybrat si z nabídky a uložit. V případě, že by třeba jednu položku ze svého menu nechtěl, nemusí měnit celé menu, nebo znovu přidávat všechny jídla. Stačí si menu otevřít podržením a jednoduše vyškrtnout, na co zrovna nemá chuť.



Obrázek . - Vlastní menu

## Objednávky a platby

O objednávku se stará další strana. Zákazník si samozřejmě může odebrat položky z objednávky. Dále mu je zobrazena celková cena objednávky. Celý proces platby už řeší integrovaná platební brána, přidána jako plugin třetí strany. Uživatel si vybere platební metodu, zadá své platební informace a odešle objednávku. Platby jsou realizovány předem kvůli tomu, aby nedocházelo k objednávkám, které si nikdo nevyzvedne. K nahlédnutí je mu také historie objednávek.



Obrázek . – Testovací platba

Obrázek . - Objednávka

# 

Obrázek . - Možnost odebrání položek z vlastního menu

# **Závěr**

Mým záměrem bylo vytvořit objednávkový systém pro restaurace rychlého občerstvení, který by mohl ušetřit čas čekání v řadě včasnou objednávkou. Zhotovil jsem dvě multiplatformní aplikace jako serverovou část a klientskou část. Ty mezi sebou komunikují přes HTTP pomocí síťové REST architektury. Server je vytvořen s využitím Mono frameworku a je schopný zpracovávat příchozí požadavky a správně na ně odpovídat. Dále se na něm dá vytvořit aktuální menu, zobrazit uživatele a historii objednávek. Klient, napsaný ve webových technologiích pomocí Ionic Frameworku, si umí vytvořit objednávku z aktuální nabídky získané od serveru zaplatit ji předem.

Budoucím vylepšením by mohlo být oddělení grafické a logické části serveru. V případě více serverů, by klient mohl lokalizovat uživatele, zobrazit mu nejbližší restauraci a cestu k ní. Dále by se mohla úplně odstranit interakce s pokladnou a to návrhem uzamykatelného termoboxu. Ten by si zákazník otevřel pomocí ID své objednávky.

Seznam použitýCH INFORMAČNÍCH ZDROJů

[1] Braintree Get Started [online]. [cit. 2017-12-18].

<https://developers.braintreepayments.com/start/overview>.

[2] Entity Framework Tutorial [online]. [cit. 2017-12-18]

<http://www.entityframeworktutorial.net/entityframework6/introduction.aspx>.

[3] How to generate a unique token which expires after 24 hours? [online]. [cit. 2017-12-19].

< [https://stackoverflow.com/questions/14643735/how-to-generate-a-unique-token-which-expires-after-24-hours /](https://stackoverflow.com/questions/14643735/how-to-generate-a-unique-token-which-expires-after-24-hours%20/)>.

[4] Ionic documentation [online]. [cit. 2017-12-18].

<https://ionicframework.com/docs/>.

[5] Ionic [2|3] Sharing Data Between Pages/Components [online].[cit. 2017-12-18].

<https://www.gajotres.net/ionic-2-sharing-data-between-pagescomponents/>.

[6] Knihovna tříd rozhraní .NET Framework: Třída HttpListener [online]. [cit. 2017-12-19].

<https://msdn.microsoft.com/cs-cz/library/system.net.httplistener(v=vs.110).aspx/>.

[7] Mono documentation [online]. [cit. 2017-12-18].

<http://docs.go-mono.com/>.

[8] NASATO, Mirko. Ionic 2 by Example: Creating Your First Ionic App [online] [cit. 2017-12-18].

<https://www.youtube.com/watch?v=qdeD9rYlBU4>.

[9] RZCODER. Node-RSA [online] [cit. 2017-12-29].

<https://github.com/rzcoder/node-rsa>.

1. Zabezpečená, jednorázově použitelná reference k platebním informacím. Umožňuje mému serveru přenášet citlivé platební informace do serveru Braintree bez přístupu k prvotním datům. [↑](#footnote-ref-1)